CPGS First Year Report

Kris Cao
Clare College

A report submitted to the University of Cambridge in partial fulfilment of the requirements for the degree of Doctor of Philosophy in Computer Science

University of Cambridge
Computer Laboratory
William Gates Building
15 JJ Thomson Avenue
Cambridge CB3 0FD
UNITED KINGDOM

Email: Kris.Cao@cl.cam.ac.uk

September 17, 2015
Contents

1 Introduction 1

2 Background and Related Work 4
  2.1 Computational analysis of narrative structure . . . . . . . . . 4
  2.2 Plot generation . . . . . . . . . . . . . . . . . . . . . . . . 6
  2.3 Computational creativity . . . . . . . . . . . . . . . . . . . 9

3 A proposal for a character-driven narrative generation system 11
  3.1 Character semantics . . . . . . . . . . . . . . . . . . . . . . 12
  3.2 Generating events . . . . . . . . . . . . . . . . . . . . . . . 12
    3.2.1 Generating interacting events . . . . . . . . . . . . . 14
  3.3 The corpus . . . . . . . . . . . . . . . . . . . . . . . . . . . 15
    3.3.1 Other resources . . . . . . . . . . . . . . . . . . . . . 16

4 Preliminary experiments 17
  4.1 Event representations . . . . . . . . . . . . . . . . . . . . . 17
    4.1.1 Experimental setup . . . . . . . . . . . . . . . . . . . 18
    4.1.2 Results . . . . . . . . . . . . . . . . . . . . . . . . . . 20
  4.2 Protagonist identification and character semantics . . . . . . . 22
    4.2.1 Corpus preparation . . . . . . . . . . . . . . . . . . . 22
    4.2.2 Character identification . . . . . . . . . . . . . . . . . 22
    4.2.3 Top-billed character prediction . . . . . . . . . . . . . 23
    4.2.4 Character semantics . . . . . . . . . . . . . . . . . . . 24

5 Plan for future years 27
List of Tables

2.1  Propp’s *dramatis personae* ................................. 5
4.1  Sample event chain ....................................... 17
4.2  Event semantics internal evaluation ...................... 20
4.3  Event semantics external evaluation ..................... 21
4.4  Top-billed character prediction accuracy ................ 23
4.5  Top-billed character prediction errors ................... 24
4.6  Some sample ‘most similar character’ predictions ........ 26
Chapter 1

Introduction

Narrative is one of the cultural universals that all humans possess (Brown, 1991). If we could teach a computer how to analyse and tell stories, this would open up a whole host of applications, such as plot-based book recommendations (De Clercq et al., 2014) or interactive plot development for computer games (Barros and Musse, 2007). As an end in itself, it would also be interesting, if only to edge closer to the day androids finally dream of electric sheep.

Central to narrative are the characters of the narrative. They, and their interactions, are the driving force of the narrative, and being able to identify and classify character types can often illuminate the type of narrative. Hence, identifying the leading characters in a story, and then assigning semantic information to them, would be of great help in analysing the story as a whole. Conversely, we can break down the task of generating a story to first generating the characters in a story, and then generating their interactions. Indeed, for dynamic plot development in computer games, one can condition character interactions and future plot events on the actions that the player character has taken so far, personalising the narrative experience according to the actions of the player.

This style of plot generation, which can broadly be termed character-centric
(Mateas and Sengers, 2003), sets up a story as the result of the interaction between autonomous agents, each with their own goals and plans. This is in contrast to author-centric systems, which seek to model the thought processes of the author, and story-centric systems, which model the structure of stories themselves. It is important to note that these are only broad classifications: an author-centric model can make use of a theory of story structure to generate narrative, just as a character-centric model might incorporate information about authorial intent.

The aim of my PhD is to develop a data-driven character-centric narrative generation system. The gist of the proposed system is to populate the story with characters, which have semantic information associated with them. With the characters in place, we can then generate interacting events for each character, conditioning on the character semantics. For instance, we can generate characters corresponding to the archetypes ‘hero’, ‘helper’, and ‘villain’, and then generate events appropriate for each archetype.

The novel idea is that we will try to discover these character archetypes in a data-driven fashion: instead of hard-coding what kinds of characters there are, we will attempt to discover character semantics from corpora of existing stories. This will reduce the amount of expert knowledge required to build a character-driven narrative system. There are many existing theories of character, and the subject can seem daunting to an outsider, with each author seemingly coming up with their own set of categories. Instead of artifically imposing character classifications, we will instead turn the problem on its head and try to see how well our semantics can predict existing classes.

Once we have our characters, we will need events for them. Inspired by Rudinger et al. (2015) and promising unpublished results on a relative of the narrative cloze task (Chambers and Jurafsky, 2008), we propose using a language model using distributionally learnt representations of events to generate events for our story. We predict that the guiding framework of the character semantics will constrain the event generator to produce a narrative that is both fluent and coherent.
We also require a suitable corpus. The ideal corpus would be a collection of short stories with a few central strong characters. We would like this corpus to be as large as possible, so that we can discover the regularities in the data. Ideally, the characters should be simple as well. After trying several alternatives, I believe that thriller films fulfil all of these criteria. Preliminary results using the top-grossing thriller film plot synopses from IMDb have given promising results.
Chapter 2

Background and Related Work

2.1 Computational analysis of narrative structure

Vladimir Propp started modern analysis of narrative in *The Morphology of the Folktale* (Propp, 1968). Analysing a collection of 100 Russian folk tales, he identified 31 plot building blocks, or *functions*, out of which one can reconstruct any of the original folk tales. He also identified 7 character classes, or *dramatis personae*, (see Table 2.1).

After Propp, researchers applied his methods to other cultures, such as Inuit folk tales (Colby, 1973). However, all of these approaches have required an expert to pore over a corpus of folk tales to tease apart the underlying structure. While computational approaches cannot aid the gathering of the source data, they can make the analysis of the data a lot easier.

Finlayson, in his PhD thesis (Finlayson, 2012), tried to automatically learn the grammar underlying the same corpus that Propp analysed. He reported some success, especially in reproducing the *Villainy, Struggle-Victory* and *Reward* functions that Propp identified, showing that in principle computational analysis of narrative is possible. However, he required extensive
<table>
<thead>
<tr>
<th>Character type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Hero</strong></td>
<td>The key character of the story. The character who undergoes the quest.</td>
</tr>
<tr>
<td><strong>Villain</strong></td>
<td>The main antagonist of the story, who struggles against the <strong>Hero</strong>.</td>
</tr>
<tr>
<td><strong>Princess</strong></td>
<td>The person (or item) that the <strong>Hero</strong> seeks. Typically the <strong>Villain</strong> is seeking to prevent the <strong>Hero</strong> in their quest.</td>
</tr>
<tr>
<td><strong>Dispatcher</strong></td>
<td>The person who sends the <strong>Hero</strong> on their quest.</td>
</tr>
<tr>
<td><strong>Helper</strong></td>
<td>Someone who aids the <strong>Hero</strong> in their quest. Sometimes the <strong>Hero</strong> has to prove themselves to the <strong>Helper</strong> first.</td>
</tr>
<tr>
<td><strong>Donor</strong></td>
<td>Someone who gives the <strong>Hero</strong> advice or a magical item to aid the <strong>Hero</strong>. Distinguished from a <strong>Helper</strong> by not being an associate of the <strong>Hero</strong>.</td>
</tr>
<tr>
<td><strong>False Hero</strong></td>
<td>Someone who shows up and tries to claim glory for the quest undertaken by the <strong>Hero</strong>. Normally exposed and punished by the conclusion of the tale.</td>
</tr>
</tbody>
</table>

Table 2.1: Propp’s *dramatis personae*

manual annotation of his corpus, including annotating the character type of each of the characters. This kind of manual annotation is expensive and hard to obtain, and it would be a step forward in computational narrative if one could automatically classify characters into their Proppian type.

In a separate strand, AI research in the 1970’s produced *scripts*, structured information about chains of events (Schank and Abelson, 1977). Designed to aid machine understanding of real-world situations, they proved to be too inflexible, requiring hand-crafting and being too domain-specific. However, recent research has opened up ways of learning scripts in an unsupervised manner. Manshadi et al. (2008) proposed learning sequences of events from a corpus of weblogs, but Chambers and Jurafsky (2008) put scripts back on the radar by automatically learning *narrative chains*, sequences of events with a shared coreferent entity.

Despite the differences in motivation, these scripts have a resemblance to the
story grammars described latterly. Both approaches aim to learn underlying event structure. However, the states in a script are a single event, and two events are considered the same if and only if they have the same main verb, while the states in a story grammar can be instantiated in any number of events (for instance many different events can give rise to the Villainy plot element). Also, while scripts tend to be single-protagonist and linear, stories typically have many interacting characters, with a branching or even cyclical event structure (if all stories had the same plot, then we would lose interest in them very quickly).

Recent research into learning typed narrative schemas has gone some way towards bridging this gap (Chambers and Jurafsky, 2009). Typed narrative schemas generalise over narrative chains by having not just a single shared argument for the events, but a member of a set of possible arguments. This allows it to learn that, for instance, ‘vampire’ and ‘zombie’ share the same type, and hence to use events it has seen for ‘zombie’ to augment the narrative schema of ‘vampire’.

## 2.2 Plot generation

Meehan’s TALE-SPIN system (Meehan, 1977) is one of the earliest plot generation systems. It is principally a problem solver: the characters have goals, and the story unfolds as the characters move towards their goals.

It, however, suffers from the same issue that afflicts all early AI research: the reliance on hand-crafted rules. The outcomes of events, how goals and events interact, and how events affect the characters are all hand-written. Many computational narrative generators since, such as MINSTREL (Turner, 1993) and MAKEBELIEVE (Liu and Singh, 2002), also suffer from this issue.

One issue with hand-crafted rules is that they are very domain-specific. This is reflected in the fact that many of these systems generate stories in a very restricted domain. TALE-SPIN generates stories about woodland creatures, MINSTREL generates stories about the Knights of the Round Table, and
Mexica (Pérez y Pérez, 1999) generates stories about the Mexicas, the pre-conquest peoples of Mexico.

While all of these generation systems have different motivations, and different structures, it is possible to find points in common. They all have some set of possible events which can happen, some way of working out the consequences of events, and some way of sequencing events to achieve an overall story. Event sequencing has usually been achieved automatically through some formal goal-solving planner, but normally rules had to be hand-written to deal with how events transformed the state of the narrative.

However, recent work has shown that some of these hand-crafted rules can be learnt automatically. Chambers and Jurafsky (2009) propose a system that can automatically learn which events follow which, and also what kinds of character can fill what events.

Further, McIntyre and Lapata (2009, 2010) have developed an automated story telling system that crucially does not rely on any human input at all. Entities, along with their potential story plots, are automatically learnt from a corpus. To generate a story, the user supplies a sentence. The system then recognises the entities in the sentence, and merges the potential plots it has learnt for those entities into a coherent and fluent whole. This demonstrates that automatic narrative generation is in principle possible. However, the system does not carry any information about the entities, beyond the name. It does not recognise that, say, as ‘dragon’ and ‘monster’ are similar, the actions they take will be similar to one another (but not identical: fire-breathing monsters are somewhat rarer). In addition, the events in a plot schema are fixed, and their system will not generate predicate-argument pairs it has not previously seen before.

Crucially, McIntyre and Lapata’s system is character-driven: the characters in the story have their own personal plots, and the plot of the story as a whole comes from the interaction of these personal plots. I believe that character-centric models have an element of simplicity which makes them attractive to study. Instead of positing theories of authorial intent, or trying to find
an underlying structure to stories, we simply need to come up with ways that actors can interact, and then let the narrative system generate the rest. In addition, with recent developments in natural language processing tools (such as coreference and named entity recognition), identifying characters in stories has become a realistic possibility. This is exciting, as it means we can automatically learn character goals and interactions, without having to hand-code rules about how characters interact. For this reason, I believe that character-centric approaches to narrative generation are the ones best suited to a data-driven analysis.

Further, by associating some kind of distributed semantics to our characters, I hope to capture a notion of character similarity, and also allow for a more fine-grained classification of character than Propp’s seven *dramatis personae*. This will mean that, as ‘monster’ and ‘dragon’ are similar as story entities, they will have similar (but not identical) possible events.

Swanson (2010) is another data-driven approach to narrative generation. He creates an interactive data-driven narrative generation system, SAY ANYTHING, which uses case-based reasoning (CBR) to continue stories. The user starts the story by inputting a sentence. SAY ANYTHING then searches its corpus for stories which are similar to the current story, and outputs 10 possible candidate sentences to continue the story. The user then picks one, and then continues the story by adding another sentence. However, the aim of the system is not to generate a story from scratch, and so it does not overlap too much with what I am trying to do.

In addition, while Propp’s morphology has directly been used as an inspiration for story-centric plot generation in Gervás (2013), it is possible that automatically learnt story grammars, such as the one presented by Finlayson (2012), can be used to generate stories, similarly to how phrase-structure grammars can be used to generate sentences. However, automatically inducing story grammars is still an underdeveloped area, and it seems technically difficult to achieve to a high degree of precision (Finlayson, 2012). In addition, there is no guarantee that the resulting stories will be coherent. For these reasons, I do not intend to pursue this line of research.
2.3 Computational creativity

More generally still, narrative generation can be seen as a branch in the field of computational creativity at large. Indeed, León and Gervás (2014) and Young (2008) explicitly use ideas from computational creativity to inform narrative generation.

Boden (1990) has modelled the creative process as a search over a conceptual space. This idea can be seen as underpinning many of the narrative generation systems detailed above, as problem solving (used by TALE-SPIN and MINSTREL) can be viewed as traversing a space of solutions looking for the optimal one. Even more concretely, McIntyre and Lapata (2010) have an explicit space of possible stories, and they are guided through this by the generation and selection processes of their genetic algorithm.

My proposed narrative generation system shares certain similarities with Boden’s model. There is a conceptual space (the space of all possible event chains), and the traversal function in this space is the narrative generation system itself. However, Boden’s model is very broad, and many different approaches to the same problem can be categorised under it. In addition, Boden’s model does not capture the difficulties in narrative generation. Once we have an adequate representation of world knowledge, we can experiment with different ways of putting stories together, but obtaining such an adequate representation has traditionally been the limiting step, and this is where the majority of labour has traditionally been expended. Indeed, Meehan (1977) explicitly has examples of ‘mis-spun’ stories where an inadequate description of the world has resulted in a badly told story.

One key facet of creativity is the creating of ideas that have never occurred before. In narrative generation, this manifests itself in both creating unseen sequences of events, and unseen events themselves. However, this has to be balanced against believability: the output of the generator cannot stray too far from what is typical. Most narrative generations skirt closer to the latter end of the axis (Gervás, 2009) – indeed, the prevalence of hand-written rules
often enforces typicality.

While a data-driven narrative generation system will naturally produce output that looks typical of the corpus it was trained on, I hope that my proposed system will be able to generate both unseen and unexpected events and unseen combinations of events. By having a compositional semantics of events, I hope that we will be able to put predicates and arguments together in novel ways, and by having a non-deterministic event generation system, I hope that we can generate novel chains of events that are still coherent.
Chapter 3

A proposal for a character-driven narrative generation system

For the reasons previously mentioned, I believe that a character-driven approach to narrative generation is the most attractive avenue to building a narrative generation system. This approach naturally breaks down into two steps:

1. Figure out how to learn representations of characters in narrative that capture their semantics (i.e. their goals, alignment, and how they interact with other characters).

2. Build a way of generating interacting event chains based on our learnt character representations.

The completed system should be able to learn a dictionary of events from a corpus, learn which events follow which by looking at ordered event co-occurrence, and then sequence events to form the basis of a narrative using a generative model trained on event sequences of previous narratives.

I also require a suitable corpus to test ideas and to train my models on. Just as Propp analysed the folk tales of his day, I believe the best corpus to use
are our modern-day folk tales: the thriller movie.

3.1 Character semantics

To start with, I would like to automatically learn semantic information about characters. This can include things like their polarity (whether they are good or bad), what actions they take and how they interact with other characters.

The idea is to build up a rich representation of characters that can replace the hand-crafted nature of previous work on character-driven plot generators. We will then use these representations as the inputs to our event generating system (see Section 3.2). A useful by-product of this work will be a way of comparing the similarity of two entities. As one of the key determiners of character semantics is the actions a character takes, there might be scope to jointly learn character semantics and event chain generation (again, see later).

A recent paper by Flekova and Gurevych (2015) aims to profile fictional characters along the introversion/extraversion scale, using their direct speech, actions, and the words used to describe that character. This is very similar to what I hope to be doing, in that we are both trying to learn some deeper representation of the character beyond the surface cues. However, while they use their representation as an end in itself, I aim to learn a representation that is less human-understandable but more suited to my task, which is event generation.

3.2 Generating events

As discussed in Section 2.2, there are already many models that learn narrative chains from text. However, I believe that by adding event semantics to event generation models, we can generate even more coherent and creative
stories, by generalising over the training data to generate novel predicate-argument pairs for events.

Recent unpublished work carried out by Mark Granroth-Wilding at Cambridge has shown that distributionally learnt representations for events can improve performance on the *multiple choice narrative cloze* (MCNC) task. The goal of this task is to predict the next event, out of a choice of five, given a chain of events so far. This has obvious applications to narrative generation: we can seed a story with some starting events, and then extend it by repeatedly picking what is most likely to happen next. I aim to use event representations similar to these as part of the building blocks of a narrative generation system: with these representations of events, we can generalise the events we generate, so we can generate previously unseen consequents of events and hopefully add a spark of novelty to our narratives.

Thus, instead of pre-learning narrative schemas and then reusing them to create new stories, I want to try and create a new story on the fly. We can have event transitions between events that have never been seen together before, and we can pair arguments with predicates that were not observed in the training data, by learning a notion of distributional similarity between events. In effect, I am proposing a language model for event generation. Language modelling to solve the narrative cloze task has already been proposed by Rudinger et al. (2015), and shown to be a promising avenue of research. I aim to show that we can go from prediction to generation successfully.

I also hope to use our learnt character semantics to guide the generation of events for a particular character. This will hopefully make sure that the actions for each character are appropriate (villains cackle, dragons breathe fire, princes marry princesses) and coherent.

This idea of generating by conditioning on external semantic information is not new. For a recent example, Bahdanau et al. (2014) approaches machine translation by encoding the source sentence into a single representation, and then generates words in the target language based on this representation. I aim to do something similar, by learning a representation of characters which
will aid generating narratives for that character.

3.2.1 Generating interacting events

Previous work so far has considered event chains for a single protagonist only. Most stories feature many characters, with interactions between the characters (conflict is a key narrative device). How we can intermingle the event chains of each character to create a coherent story as a whole is a thorny issue.

McIntyre and Lapata (2010) propose just merging the plot schemas for each entity whenever there is an event that features both entities in opposite roles. However, if our generative model for *James Bond* produces the event ‘punch *Q*’, this will obviously intrude on the event chain of *Q*. At what point should we schedule this interruption? It could be the case that *Q* is never near enough to *Bond* to be punched. This suggests the need for some sort of global control.

Pichotta and Mooney (2014) propose an extension of narrative script learning which models interactions between entities, and learns one narrative chain per document which contains all the entities in that document, rather than one narrative chain per entity per document. Their model then estimates probabilities for unseen events taking into account both the verb of the event and the coreferent arguments it shares with the existing script. This is a natural extension to the narrative script model, and seems like it will solve many of the issues that have just been highlighted.

Finally, Méndez et al. (2014) have modelled character affinity for agent-based narrative generation. For each pair of characters, there is an asymmetric affinity value, which constrains the interactions those two characters can have. In addition, if one character proposes an interaction with another, and this proposal is accepted, then their affinity increases, and vice-versa if the proposal is rejected. While situated in a different approach to narrative generation, this bears strong parallels to what I want to do: I also want
to learn the ‘affinity’ between different kinds of characters, and how this constrains how they interact.

3.3 The corpus

Data-driven approaches to natural language processing require the use of a suitable corpus of discovery. Propp used a collection of traditional folk tales, which was very well-suited to his task: folk tales tend to be formulaic, with a few central protagonists who can be classified easily, and the plots are simple and linear. McIntyre and Lapata used a collection of fairy tales, which share many of the same properties of folk tales. However, a crucial issue is data paucity. Modern machine learning approaches have been found to work best with large amounts of data, and unfortunately there are only a limited number of folk tales. Thus, the ideal corpus to study computational narrative ought to combine the best qualities of the existing corpora with easy availability and quantity.

After trying a few different resources, I believe that condensed plot synopses of thriller films is a promising corpus to use. Thriller movies tend to have strong character types, with a simple story of good against evil. I experimented with both the highest-rated and the top-grossing movies, and I found that typically top-grossing movies have simpler plots and more archetypal characters. For the majority of my preliminary experiments, I used a corpus consisting of the top 200 US grossing thriller movies. These contained multiple films in the same franchise, which is beneficial for experiment validation - we expect the character of James Bond to not change much across different Bond movies, for instance.

However, film is primarily a visual medium. While scripts convey dialogue, it is not as easy to get a sense of narrative by reading them, compared to reading a short story. Fortunately, IMDb features user-written plot synopses for films, which give a summary of the storyline of the film, including spoilers. I was able to mine this as a resource to build my corpus. For later valida-
tion purposes, I will extract plot synopses from a different resource (such as Wikipedia, say) and check that the output is similar.

Films offer the additional benefit of a cast list. This features the characters in the film, usually in order of importance. The list itself can be used to correct the output of an NER and co-reference tool, allowing automatic annotation of entity and coreference. The ordering can be used to evaluate the output of algorithms predicting how important a character is to the story.

3.3.1 Other resources

I also tried using action movies to build a corpus, but one central issue with action movies is that many modern action movies tend to feature an ensemble cast and it is difficult to identify who the most important characters are in such movies. This makes it hard to issue annotator guidelines for labelling the characters with their character roles (for instance, where does the line between _Hero_ and _Helper_ lie?). Moreover, a _Hero_ is usually the protagonist of the narrative, but if there are multiple protagonists, does this make them all _Heros_? For these reasons, I decided to use thriller movies, which usually have a clear single protagonist and a clear single antagonist.

Another resource I tried was summaries of Shakespeare plays. Unfortunately, these have the combined issue of data paucity and ambiguous characterisation. For these reasons, I did not pursue this as a data source.
Chapter 4

Preliminary experiments

4.1 Event representations

**Text:** Robbers made a big score, fleeing after stealing more than $300,000 from Wells Fargo armored-truck guards who were servicing the track’s ATMs, the Police Department said. The two Wells Fargo guards reported they were starting to put money in the clubhouse ATM when a man with a gun approached and ordered them to lie down...

**Predicate events:** service($x_0$, ATMs), report($x_0$), put($x_0$, money, in_clubhouse), order(man, $x_0$), lie_down($x_0$)

**word2vec ‘sentence’:** service, report, put, order, lie_down

**word2vec ‘sentence’ with arguments:** service, arg:ATMs, report, put, arg:money, arg:man, order, lie_down

Table 4.1: Example event chain extraction. Mentions of an entity $x_0$ (highlighted in red) are identified by coreference resolution. Events are then extracted from verbs to which an entity is an argument (highlighted in blue). The extracted chain is then shown below, as well as the ‘sentences’ that were presented to word2vec.

I first investigated how best to capture the semantics of events in an event
chain. An event is taken to be a predicate-argument combination where the main predicate is a verb, and an event chain is a sequence of events that share a coreferent argument (called the protagonist). These are a slight modification of the notions introduced in Chambers and Jurafsky (2008) in the context of learning narrative chains, who define an event as a tuple of a verb and the syntactic dependency of the verb with the protagonist.

However, for the purposes of narrative generation, it makes sense to also include the other arguments to the event verb. Indeed, if we want the event chains of our entities to interact with each other, we are forced to do this, to capture the intertwining nature of the narrative.

With this and my remarks from Section 3.2 in mind, I set out to learn and evaluate a simple a distributional representation of events. While this is not directly relevant to my goal of narrative generation, I believe that having a good distributional event semantics will aid in narrative generation.

4.1.1 Experimental setup

The New York Times portion of the Gigaword corpus comprises all the articles from the NYT between 1994-2004. Using Mark Granroth-Wilding’s parsed and co-reference annotated version of the Gigaword corpus, I set out to learn representations for these events. I extracted all the event chains from Gigaword, then used a variety of models to obtain vector representations for these events. By treating these event chains as contexts for the events they contain, I could use standard tools from distributional semantics, such as positive pointwise mutual information (PPMI), latent semantic analysis (LSA) and word2vec (Mikolov et al., 2013a) to get vectors for events. My initial models all used the verb in an event as a proxy for the event as a whole.

Pointwise mutual information, or PMI, replaces raw co-occurrence counts of tokens $x$ and $y$ with the pointwise mutual information between them. This is positive if $x$ and $y$ occur together more often than expected, 0 for independence and negative if $x$ and $y$ occur less often than expected. Positive
PMI (or PPMI) replaces all negative PMI values with 0. For the PPMI model of event semantics, I calculated PPMI-weighted co-occurrence vectors for each event. I disregarded any event that had been seen fewer than 1000 times in the event chains (note this is not the same as being seen fewer than 1000 times overall, as many verbs are not part of any event chain).

I then applied singular value decomposition to the PPMI-weighted co-occurrence matrix. This factorises a matrix as $M = U\Sigma V^*$, where $\Sigma$ is the singular values of $M$. We can truncate $\Sigma$ keeping only the top $N$ singular values as a way of reducing the dimensionality of $M$ and obtain dense vector representations. For the SVD model, I used $N = 100$.

Finally, I used the word2vec software from Mikolov et al. (2013a) to learn dense representations of events. This learns vectors for words which aim to predict the words in their context. I tested both the CBOW and Skipgram models of word2vec, and kept the other parameters as default.

To evaluate the vectors that I obtained, I used the verb-object part of the Mitchell and Lapata phrase similarity dataset (Mitchell and Lapata, 2010). This consists of 108 verb-object pairs, together with 14 human-annotated similarity scores for each verb-object pair, which I averaged. To get a model prediction for the similarity of two verb-object pairs, I used the verb of each pair and calculated the cosine of the vectors the model had learnt for that verb. The results are in Table 4.2. The evaluation was the Spearman rank correlation between the model similarity scores and the human similarity scores.

After these preliminary results, I set out to compare the performance of my models to existing models of verb similarity, as well as including some form
of composition to them so that argument information could be taken into account. As the best-performing candidate, I tested the CBOW vectors that \texttt{word2vec} had learnt from the event chain contexts (hereafter \texttt{event2vec}) against the 300-dimensional vectors $W$ trained by Mikolov et al. (2013b) and made available by the authors, which we will call \texttt{Mikolov-verb}. Both these models use only the verb of each verb-object pair to compute the similarity of the pair.

In addition, I also tried to add event arguments to the \texttt{event2vec} model. I learnt representations for arguments by including them in the event chains (see Table 4.1 for details). The \texttt{event2vec+arg} model forms the representation of a verb-object pair by summing up the vectors for the verb and the object. Similarly, the \texttt{Mikolov-verb+arg} model sums together the vectors from $W$ for the verb and the object.

We used the best performing verb-object model in Polajnar et al. (2014) as a baseline. Their method constructs vectors using tTest weighted co-occurrence, and then tests various methods of compositionality. They report best results in the verb-object similarity task by summing these vectors together. As they calculate the Spearman rank correlation without averaging the human similarity scores, the results in Table 4.3 are against the raw Mitchell and Lapata dataset.

### 4.1.2 Results

<table>
<thead>
<tr>
<th>Model</th>
<th>$\rho$</th>
</tr>
</thead>
<tbody>
<tr>
<td>PPMI</td>
<td>0.405</td>
</tr>
<tr>
<td>SVD100</td>
<td>0.438</td>
</tr>
<tr>
<td>\texttt{word2vec} Skipgram</td>
<td>0.516</td>
</tr>
<tr>
<td>\texttt{word2vec} CBOW</td>
<td>0.521</td>
</tr>
</tbody>
</table>

Table 4.2: Results summary for initial model evaluation. Human similarity scores have been averaged.

Both the \texttt{event2vec} and the \texttt{Mikolov-verb} models beat Polajnar et al.
Table 4.3: Results summary for external model comparison. Human similarity scores have not been averaged in predicting the similarity of verb-object pairs.

More interestingly, EVENT2VEC and MIKOLOV-VERB models perform comparably in a verb-only setting, despite MIKOLOV-VERB being trained using orders of magnitude more data (300 billion tokens vs 12 million). However, MIKOLOV-VERB+ARG performs a lot better than EVENT2VEC+ARG. One reason may be that we are not capturing argument semantics very well using event chain contexts. Despite this disparity, I do not believe that event representations of this style are worse than using pre-trained window-based vectors.

Recent work by Mark Granroth-Wilding has shown that in the multiple choice narrative cloze task, a downstream task closely related to narrative generation, models similar to EVENT2VEC+ARG perform far better than using pre-canned word2vec vectors. In addition, more sophisticated ways of building representations of events by composing the verb and its arguments have shown incredible promise in this task. I intend to use event representations similar to these as part of my work on building a narrative generation system.

<table>
<thead>
<tr>
<th>Model</th>
<th>$\rho$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Polajnar et al. (2014)</td>
<td>0.33</td>
</tr>
<tr>
<td>EVENT2VEC</td>
<td>0.35</td>
</tr>
<tr>
<td>EVENT2VEC+ARG</td>
<td>0.39</td>
</tr>
<tr>
<td>MIKOLOV-VERB</td>
<td>0.36</td>
</tr>
<tr>
<td>MIKOLOV-VERB+ARG</td>
<td>0.44</td>
</tr>
</tbody>
</table>
4.2 Protagonist identification and character semantics

I performed some small-scale experiments on a cut-down version of the thriller movie corpus to test how well I could automatically identify the main characters. I also tested some simple schemes to associate some semantics to these characters.

4.2.1 Corpus preparation

The initial corpus consisted of all of the highest 200 US grossing thriller films that had plot synopses on IMDb. This consisted of 185 documents, with 394,534 tokens. These documents were then tokenised and processed, using the C&C tools for PoS tagging and dependency parsing and OpenNLP for named entity recognition and coreference resolution. At the same time, I also obtained the cast list for each film, which identifies the leading characters in the film.

4.2.2 Character identification

Vala et al. (2015) considered the problem of identifying characters in literature. They use an eight-stage pipeline approach to link potential mentions of the same entity and then delink mentions of separate entities, using inferred semantic information about the referents (gender, first name and honorific). However, their task is complicated as they do not have access to a gold-standard list of characters. I have access to film cast lists, so I was able to sidestep many of their issues and use a simple heuristic to match the entities and mentions OpenNLP gave to characters in the cast list.

First, for each entity that OpenNLP identified, I constructed a normal form for each entity by taking the shortest non-pronoun mention. Then, for each entity, I found the longest string overlap between the normal form of the
Table 4.4: Top 1 and Top 3 accuracy for the top-billed character prediction task

<table>
<thead>
<tr>
<th>Model</th>
<th>Top 1 accuracy</th>
<th>Top 3 accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>First mention</td>
<td>0.546</td>
<td>0.843</td>
</tr>
<tr>
<td>Most mentions</td>
<td>0.645</td>
<td>0.865</td>
</tr>
<tr>
<td>Longest event chain</td>
<td>0.681</td>
<td>0.870</td>
</tr>
</tbody>
</table>

entity and each character. As many entities discovered by the OpenNLP NER system do not correspond to characters, I matched an entity to the character it shared the longest common substring with only if this substring was a subword of both the entity normal form and the character name.

### 4.2.3 Top-billed character prediction

The first task I tried was to predict the top-billed characters in the cast list. The cast list typically (but not always) lists characters in order of importance, so we can use this to evaluate algorithms that aim to identify the main characters in a story and rank them in order of importance.

I tested various schemes, such as order of first mention in the text, number of mentions, and length of event chain. The results were evaluated based on either top-1 accuracy (whether the actual top-billed character matched the predicted top-billed character) and top-3 accuracy (whether the actual top-billed character was in the highest 3 predictions for the top-billed character). The results are summarised in Table 4.4.

From the results, it appears that simple unsupervised approaches to the problem of identifying the top-billed character do remarkably well. The best models have a top-3 accuracy of about 85%, and even have impressive top-1 accuracies. As I am using top billing as a proxy for the most important character, this shows that automatically identifying the protagonist of a story is a solvable issue.
A close examination of the top-1 result errors for the ‘longest event chain’ model reveals some interesting results. A few films bill the cast in alphabetical or appearance order. Some franchise films put a central character in the franchise as the top bill for all movies in the franchise, even if another character is more important in a specific movie. Finally, the model does sometimes just make the wrong prediction, even when the top-billed is the most important character. It is in this category that improvement can be made, perhaps by learning a better understanding of the deeper semantics of each character, and so distinguishing between true heroes and accomplices, which seems to be the major source of error.

### 4.2.4 Character semantics

I also tried to associate some simple semantics to the characters that were automatically identified. One starting attempt was to classify characters based on their actions.

Using the extracted event chains from the top-billed character prediction task, I built a vector representation for each character by averaging all of the vectors of the events in that character’s event chain. As a starting point,
I represented each event using the verb of that event, and used the vectors from Mikolov et al. (2013b).

To measure how similar two character embeddings were, I used a variety of models: dot product similarity between two vectors, cosine similarity, and Euclidean distance. Some sample ‘most similar’ predictions are shown in Table 4.6.

The preliminary results looked promising, particularly the predictions cosine and Euclidean gave. The reason they are so similar is that the character representations are in effect normalised, so that if two vectors are close in cosine similarity, they are likely to be close together geometrically as well.

However, I would like to run some objective evaluation on the quality of these representations. One possible evaluation scheme is to use the fact that there are many films from the same franchise in my corpus, and evaluate how many instances of the same character from other films the prediction system outputs.

In addition, the predictions of similarity the model makes are sometimes still a little puzzling. I would like to try more sophisticated techniques of learning character semantics. The paragraph vector method of Le and Mikolov (2014) seems an interesting idea to try: they learn representations of documents that aim to predict words in that document.
### Table 4.6: The predicted top 5 most similar characters to each given character

<table>
<thead>
<tr>
<th>Character</th>
<th>DOT-PROD</th>
<th>COSINE</th>
<th>EUCLIDEAN</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>James Bond (Die Another Day)</strong></td>
<td>Felix Cortez</td>
<td>Barney Ross</td>
<td>Barney Ross</td>
</tr>
<tr>
<td></td>
<td>(Clear and Present Danger)</td>
<td>(The Expendables)</td>
<td>(The Expendables)</td>
</tr>
<tr>
<td></td>
<td>Danny Rivetti</td>
<td>Ethan Hunt</td>
<td>Ethan Hunt</td>
</tr>
<tr>
<td></td>
<td>(Crimson Tide)</td>
<td>(Mission: Impossible)</td>
<td>(Mission: Impossible)</td>
</tr>
<tr>
<td>Owen Davian (Mission: Impossible III)</td>
<td>James Bond</td>
<td>Ethan Hunt</td>
<td>Ethan Hunt</td>
</tr>
<tr>
<td></td>
<td>(The World Is Not Enough)</td>
<td>(Mission: Impossible)</td>
<td>(The World Is Not Enough)</td>
</tr>
<tr>
<td></td>
<td>(Ocean’s Twelve)</td>
<td>(The Departed)</td>
<td>(The Departed)</td>
</tr>
<tr>
<td></td>
<td>Danny Rivetti</td>
<td>Ethan Hunt</td>
<td>Ethan Hunt</td>
</tr>
<tr>
<td></td>
<td>(Clear and Present Danger)</td>
<td>(Mission: Impossible)</td>
<td>(Mission: Impossible)</td>
</tr>
<tr>
<td></td>
<td>Ethan Hunt</td>
<td>Colin Sullivan</td>
<td>Colin Sullivan</td>
</tr>
<tr>
<td></td>
<td>(Mission: Impossible III)</td>
<td>(The Departed)</td>
<td>(The Departed)</td>
</tr>
<tr>
<td></td>
<td>Danny Rivetti</td>
<td>Ethan Hunt</td>
<td>Ethan Hunt</td>
</tr>
<tr>
<td></td>
<td>(Clear and Present Danger)</td>
<td>(Mission: Impossible)</td>
<td>(Mission: Impossible)</td>
</tr>
<tr>
<td></td>
<td>Ethan Hunt</td>
<td>Colin Sullivan</td>
<td>Colin Sullivan</td>
</tr>
<tr>
<td></td>
<td>(Mission: Impossible III)</td>
<td>(The Departed)</td>
<td>(The Departed)</td>
</tr>
<tr>
<td>Jim Phelps (Mission: Impossible)</td>
<td>Flattop (Dick Tracy)</td>
<td>Reynard (Rush Hour 3)</td>
<td>Reynard (Rush Hour 3)</td>
</tr>
<tr>
<td></td>
<td>John Rooney</td>
<td>Reynard (Rush Hour 3)</td>
<td>Sloan (Wanted)</td>
</tr>
<tr>
<td></td>
<td>(Road to Perdition)</td>
<td>Reynard (Rush Hour 3)</td>
<td>Sloan (Wanted)</td>
</tr>
<tr>
<td></td>
<td>Felix Cortez</td>
<td>Sloan (Wanted)</td>
<td>Sloan (Wanted)</td>
</tr>
<tr>
<td></td>
<td>(Clear and Present Danger)</td>
<td>(Wanted)</td>
<td>(Wanted)</td>
</tr>
<tr>
<td></td>
<td>Alex West</td>
<td>Blackheart (Ghost Rider)</td>
<td>Blackheart (Ghost Rider)</td>
</tr>
<tr>
<td></td>
<td>Lara Croft: Tomb Raider</td>
<td>Blackheart (Ghost Rider)</td>
<td>Blackheart (Ghost Rider)</td>
</tr>
<tr>
<td></td>
<td>Benjamin Dunn</td>
<td>Claire Phelps</td>
<td>Claire Phelps</td>
</tr>
<tr>
<td></td>
<td>(Mr. &amp; Mrs. Smith)</td>
<td>(Mission: Impossible)</td>
<td>(Mission: Impossible)</td>
</tr>
<tr>
<td></td>
<td>Jim Phelps</td>
<td>John Patrick Mason</td>
<td>John Patrick Mason</td>
</tr>
<tr>
<td></td>
<td>(Mission: Impossible)</td>
<td>(The Rock)</td>
<td>(The Rock)</td>
</tr>
<tr>
<td>Lara Croft (Lara Croft: Tomb Raider)</td>
<td>Jeff Hendricks</td>
<td>Richard Kimble</td>
<td>Richard Kimble</td>
</tr>
<tr>
<td></td>
<td>(Jaxx 2)</td>
<td>(The Fugitive)</td>
<td>(The Fugitive)</td>
</tr>
<tr>
<td></td>
<td>Lily Raines</td>
<td>Frank Moses (RED)</td>
<td>Frank Horrigan (In the Line of Fire)</td>
</tr>
<tr>
<td></td>
<td>(In the Line of Fire)</td>
<td>(RED)</td>
<td>(RED)</td>
</tr>
<tr>
<td></td>
<td>Harlee Shane</td>
<td>Frank Horrigan</td>
<td>Frank Horrigan</td>
</tr>
<tr>
<td></td>
<td>(The Towering Inferno)</td>
<td>(In the Line of Fire)</td>
<td>(In the Line of Fire)</td>
</tr>
<tr>
<td></td>
<td>Shariff Gravestay</td>
<td>Bryan Mills</td>
<td>Bryan Mills</td>
</tr>
<tr>
<td></td>
<td>Dr. Cawley (Shutter Island)</td>
<td>(Taken)</td>
<td>(Taken)</td>
</tr>
<tr>
<td></td>
<td>Jane Smith (Mr. &amp; Mrs. Smith)</td>
<td>Bryan Mills</td>
<td>(Taken)</td>
</tr>
<tr>
<td></td>
<td>Dr. Cawley (Shutter Island)</td>
<td>(Taken)</td>
<td>(Taken)</td>
</tr>
<tr>
<td></td>
<td>Bryan Mills</td>
<td>Dr. Cawley (Shutter Island)</td>
<td>(Taken)</td>
</tr>
<tr>
<td></td>
<td>(Taken)</td>
<td>(Taken)</td>
<td>(Taken)</td>
</tr>
</tbody>
</table>

26
Chapter 5

Plan for future years

While the preliminary results show that there is promise for analysing characters in films using NLP tools, and in learning the semantics of events, there is clearly still much to be done before the dream of a character-based narrative generation system can be realised. Here are some intermediate challenges that I would like to solve, along with a rough time plan:

1. **Learn additional character semantics.** (Oct 2015-Mar 2016)
   Actions alone seem insufficient to give a well-rounded view of many characters, as many actions are shared by both heroes and villains. I would like to identify how to identify the ‘polarity’ of characters – that is, whether they are good or bad. This is superficially similar to the sentiment analysis task, and similar approaches may work. However, the problem seems different enough that I cannot expect to use off-the-shelf tools to solve it.

   This is possibly one area where better event semantics will help (‘kill child’ has different connotations to ‘kill henchman’, for instance). I will see if, by having a better understanding of the events a character performs, we can better classify them into ‘good’ or ‘evil’. There may also be other factors, such as how the character is described, and even the name of the character.
2. **Write up polarity assignment work** (Apr 2016-Jun 2016)
   I will write a paper based on my work on the polarity assignment task, aiming to submit to EMNLP.

3. **Predicting events conditioning on character semantics** (Jul 2016-Sep 2016)
   At this stage, I should have some representation of characters that captures some part of their essence. The next step is to see whether these representations help the event prediction task. I will aim to develop a scheme that uses information about the character to help predict what’s going to happen next.

4. **Add character interactions to the event prediction task** (Oct 2016-Dec 2016)
   So far I have only considered characters in isolation. However, it seems like it would be helpful to consider interactions between characters. Can we automatically learn that if we have a hero and a villain in the same story they are likely to be in conflict?

5. **Go from a prediction system to a generative system** (Jan 2017-Jun 2017)
   So far, I have only focussed on getting good at predicting what happens next. Now, I would like to develop the system to be able to generate a cohesive story.

   The issue with just taking the highest prediction of what happens next each time is that we can become ‘stuck’ in a certain state and not move the narrative forwards. We may need to learn some kind of underlying structure for stories, and generate events to fill in this structure.

6. **Writing up** (Jul 2017-Sep 2017)
   I will leave the last few months of my PhD to write up my results.
Bibliography


